RELEVANCIA Y REDUNDANCIA EN SISTEMAS DE CLASIFICACIÓN DIFUSA.

A. del Amo
Dpto. Estadística e I.O
Facultad de Matemáticas
Universidad Complutense
ana_amo@mat.ucm.es

J. Montero
Dpto. Estadística e I.O
Facultad de Matemáticas
Universidad Complutense
javier_montero@mat.ucm.es

G. Biging
Dept. of Environmental Sciences
Berkeley University
California U.S.A
biging@nature.Berkeley.edu

Resumen

En este trabajo presentamos un enfoque de sistemas de clasificación difusa basado en modelos agregativos, de forma que la partición de Ruspini aparece como una solución aditiva particular. A partir de la definición de sistema de clasificación difusa recursivo, estudiamos los conceptos de relevancia y redundancia. Utilizaremos estos conceptos en un ejemplo de clasificación espectral de una imagen digital.

Palabras Clave: Partición de Ruspini, Sistema recursivo, Relevancia, Redundancia.

1 INTRODUCCIÓN

Clasificación y control han sido desde los comienzos de la historia de la teoría de conjuntos difusos (ver, el trabajo de Zadeh [18]), los dos campos de mayor desarrollo práctico (ver, entre otros los trabajos de Bezdek [4], Zadeh [19]). De hecho, dentro de cada una de estas dos áreas, muchos problemas fácilmente sugieren la aplicación de conceptos difusos. Algunas veces, una aproximación difusa parece ofrecer una simplificación de una realidad extremadamente compleja, como ocurre en problemas de control; pero en otras ocasiones, los conceptos que el decisor tiene en su mente son difusos por naturaleza, en el sentido de que admiten grados de verificación, como en muchos problemas de clasificación, donde la introducción de clases nítidas representan una simplificación de la realidad excesiva, que nos llevará a interpretaciones erróneas de las observaciones reales.

En muchas aplicaciones de modelos de clasificación difusa se parte de una familia de clases $\mathcal{C}$ previamente definidas. La cuestión es entonces determinar para cada objeto x bajo consideración, el grado $\mu_c(x)$ en que el objeto x pertenece a la clase c. Para ello, se tiene definida una función de pertenencia

$$\mu_c : X \to [0,1]$$

para cada clase $c \in \mathcal{C}$ (ver, entre otros, el trabajo de M. Roubens [11]).

Desde nuestro punto de vista, esta aproximación resulta todavía bastante poco realista, ya que muchos decisores encontrarán serias dificultades para asignar esos grados de pertenencia a una clase sin tener en cuenta las otras posibilidades de clasificación. Todo método de clasificación es en la práctica altamente dependiente de la familia (cerrada) que el decisor está forzado a considerar, incluso en el contexto nítido, donde el decisor suele revisar todas las clases antes de escoger una clase concreta como la más apropiada.

Un concepto clave dentro de clasificación es, por lo tanto, la noción de *partición*, tanto en cuanto nos aporta una familia estructurada de clases dentro de la cual hemos de movernos (las clases están fuertemente interrelacionadas entre sí).

La noción de *partición difusa* fue introducida por Ruspini [12] (ver también el artículo de J.C. Bezdek y J.D. Harris [3]) y supone que dada una familia discreta $\mathcal{C}$ de *clases*, se asume que para cada objeto $x \in X$ bajo consideración se verifica que

$$\sum_{c \in \mathcal{C}} \mu_c(x) = 1.$$

Cada objeto pertenece hasta un cierto punto a cada clase, y el total de pertenencia se distribuye entre todas las clases. De esta manera se generalizaba trivialmente el concepto de partición nítida donde la función de pertenencia es tal que: $\mu_c(x) \in \{0,1\}$, $\forall x, \forall c$, de forma que cada objeto estará en una y sólo una clase:

$$\forall x \in X \Rightarrow \exists c \in \mathcal{C} \quad / \quad \mu_c(x) = 1, \mu_k(x) = 0, \forall k \neq c.$$

La propuesta inicial de Ruspini representa, desde nuestro punto de vista, una situación deseable en muchas

ocasiones, pero aún así todavía muy restrictiva en la práctica de la modelización difusa. Con frecuencia, las clases difusas no definen una partición de Ruspini, y es sólo a través de un largo y tedioso proceso de aprendizaje, cuando el decisor puede realmente definir una partición difusa en el sentido de Ruspini, de forma que cada objeto se encuentra plenamente determinado y sin rastro de información superflua.

Además, ese modelo *ideal* de clasificación en el sentido de Ruspini puede ser no sólo imposible de conseguir, sino que además, pueden existir situaciones en las que no sea ni siquiera deseado. Así, por ejemplo, en algunos problemas de clasificación de frutas, las restricciones impuestas por el mercado son un número muy grande de clases *solapadas*, de tal modo que una pieza de fruta puede estar simultáneamente asociada a varias clases con un grado de pertenencia próximo o igual a uno.

Algunas de las dificultades que presenta la noción de partición de Ruspini pueden ser parcialmente resueltas con la aproximación más débil propuesta por algunos autores (ver entre otros, H. Thiele [13, 14]). Sin embargo, nosotros proponemos aquí representar los sistemas de clasificación difusa a través de un modelo agregativo, donde el modelo de E. Ruspini no será más que un modelo aditivo particular.

2 PROCESO DE AGREGACIÓN

Uno de los principales objetivos en un problema de clasificación es la agregación de la información obtenida para cada uno de los objetos, así como de las clases. Es necesario encontrar operadores capaces de agregar información de diferentes clases e incluso diferentes dimensiones. La asociatividad permite agregar información de cualquier conjunto de datos independientemente de su dimensión, a partir de operadores binarios. Veamos como utilizando reglas recursivas es posible agregar información de cualquier conjunto de datos sin tener que imponer asociatividad (ver entre otros [1] y [5]). En primer lugar introducimos la definición de regla recursiva.

Definición 1. *Una regla recursiva por la izquierda es una familia de operadores*

$$\{\phi_n : [0,1]^n \to [0,1]\}_{n>1}$$

tal que existe una secuencia de operadores binarios

$$\{L_n : [0,1]^2 \to [0,1]\}_{n>1}$$

tal que

$$\begin{aligned} \phi_2(a_1,a_2) &= L_2(\pi(a_1),\pi(a_2)), \\ \phi_n(a_1,\ldots,a_n) &= L_n(\phi_{n-1}(a_{\pi(1)},\ldots,a_{\pi(n-1)}),a_{\pi(n)}), \end{aligned}$$

$\forall n > 2$ *y para alguna regla de ordenación previa* π *([1, 5]).*

La noción de recursividad por la derecha será análoga.

Definición 2. *Una regla recursiva por la derecha es una familia de de operadores*

$$\{\phi_n : [0,1]^n \to [0,1]\}_{n>1}$$

tal que existe una secuencia de operadores binarios

$$\{R_n : [0,1]^2 \to [0,1]\}_{n>1}$$

tal que

$$\begin{aligned} \phi_2(a_1,a_2) &= R_2(a_{\pi(1)},a_{\pi(2)}), \\ \phi_n(a_1,\ldots,a_n) &= R_n(a_{\pi(1)},\phi_{n-1}(a_{\pi(2)},\ldots,a_{\pi(n)})), \end{aligned}$$

$\forall n > 2$ *y para alguna regla de ordenación previa* π.

Definición 3. *Una* regla recursiva *es aquella regla tal que las representaciones por la izquierda y por la derecha se pueden obtener con una misma regla de ordenación previa.*

La recursividad se verifica cuando

$$\begin{aligned} \phi_n(a_1,\ldots,a_n) = R_n(a_{\pi(1)},\phi_{n-1}(a_{\pi(2)},\ldots,a_{\pi(n)})) = \\ = L_n(\phi_{n-1}(a_{\pi(1)},\ldots,a_{\pi(n-1)}),a_{\pi(n)}) \end{aligned}$$

Teorema 1. *Sea* $\{\phi_n\}_{n>1}$ *una regla recursiva estándar regular (ver [1]),* $N : [0,1] \longrightarrow [0,1]$ *una función negación. Entonces* $\{\varphi_n : [0,1]^n \longrightarrow [0,1]\}_{n>1}$ *tal que para todo* $(a_1,\ldots,a_n) \in [0,1]^n$:

$$\varphi_n(a_1,\ldots,a_n) = N^{-1}(\phi_n(N(a_1),\ldots,N(a_n)))$$

es una regla recursiva estándar regular.

Utilizaremos estos resultados para la definición de sistema de clasificación difusa recursivo, a partir del cual estudiaremos los conceptos de relevancia y redundancia.

3 SISTEMA DE CLASIFICACIÓN DIFUSA RECURSIVO

Supongamos un conjunto finito de objetos X. Llamaremos *sistema de clasificación recursivo* a una familia finita C de clases difusas, donde cada $c \in C$ tiene una función de pertenencia asociada

$$\mu_c : X \to [0,1]$$

junto con la tripleta que llamaremos *tripleta recursiva* $(\{\phi_n\},\{\varphi_n\},N)$, donde

- $\phi = \{\phi_n\}$ es una regla recursiva estándar en el papel de función disyunción,

- $N : [0,1] \rightarrow [0,1]$ es una función negación, i.e., función continua, estrictamente decreciente, asociativa tal que $N(0) = 1$ y $N(1) = 0$ y
- $\varphi = \{\varphi_n\}$ regla recursiva estándar en el papel de función conjunción definida como sigue:

$$\varphi_n(a_1, \ldots, a_n) = N^{-1}(\phi_n(N(a_1), \ldots, N(a_n))).$$

3.1 Relevancia

En general, cuando se trabaja con sistemas, de cualquier naturaleza, han de tenerse en cuenta los elementos que lo componen. En el caso en que trabajemos con sistemas de clasificación, puede considerarse que los elementos o componentes del sistema son las clases. En este sentido, diremos que una clase es *relevante* cuando al prescindir de la misma como componente del sistema de clasificación, al menos una de las unidades u objetos queda de alguna manera indeterminado. Es *irrelevante* en caso contrario, es decir, cuando al prescindir de ella ningún objeto queda indeterminado. Así, por ejemplo, una clase *vacía* c tal que

$$\mu_c(x) = 0 \qquad \forall x \in X$$

debiera ser eliminada de nuestro modelo, ya que ningún elemento pertenece a ella, es decir, sería una clase irrelevante. Lo mismo ocurre cuando dos de las clases están completamente replicadas, es decir, cuando

$$\mu_c(x) = \mu_k(x), \qquad \forall x \in X,$$

siendo c, k dos clases diferentes ($c \neq k$). Entonces, ambas clases son iguales y podemos eliminar una ya que al menos una de ellas es irrelevante. Estas dos situaciones *nítidas* pueden ser resueltas fácilmente, simplemente eliminando algunas clases tras una trivial comparación. La dificultad aparece cuando nos enfrentamos a situaciones de alguna manera *cercana* a esas dos situaciones extremas. En ambos casos, aparecen clases *casi* irrelevantes, en el sentido de que apenas nos ayudan a lograr una mejor clasificación.

H. Thiele en los artículos [13, 14], excluye del modelo aquéllas clases *vacías*. Nuestro modelo también debiera eliminarlas y tener en cuenta sólo las clases c que sean *relevantes* en el sentido de que $\mu_c(x) > 0$ para algún objeto x. Pero la *relevancia* también admite una cierta gradación y una primera propuesta para medir la *relevancia* de la clase $c \in \mathcal{C}$ puede obtenerse a través del índice

$$\phi\{\mu_c(x)/x \in X\}$$

de forma que se puede decir que una clase c es menos *importante* en la medida en que dicho valor es más pequeño. Una clase será menos relevante cuanto menor sea dicho índice. Pero esta argumentación tiene sus riesgos:

- $\mu_c(x)$ puede ser baja, pero aún así puede constituir la única información acerca del objeto x basta considerar el caso en que $\mu_c(x) > 0$ y $\mu_k(x) = 0, \forall k \neq c$.
- $\mu_c(x)$ puede ser alto, pero aún así no darnos ninguna información diferenciadora si, por ejemplo, $\mu_c(x) = \mu_c(y), \forall y \in X$.
- $\mu_c(x)$ puede ser alto, pero puede ocurrir que dicho objeto puede ser descrito mucho más eficazmente por otras clases: por ejemplo, cuando $\mu_k(x) > \mu_c(x)$, $\mu_k(y) = \mu_c(y), \forall y \neq x$.

En principio, la *relevancia* debiera evaluarse no para cada clase aislada c, sino que deben considerarse familias de clases dentro de $\mathcal{C}$ y plantearse si dichas familia deben ser *mantenidas* en el modelo. Para una clase c, por ejemplo, habrá que comparar, para cada objeto x,

$$\phi\{\mu_c(x)/\, k \in \mathcal{C}\} \tag{1}$$

$$\phi\{\mu_k(x)/\, k \in \mathcal{C}, k \neq c\} \tag{2}$$

En este caso particular podemos resumir el estudio de la relevancia en los modelos de clasificación de la siguiente forma. Compararemos para cada objeto las dos cantidades expuestas anteriormente, que representarán cómo está explicado un objeto dentro de una clase, frente al resto de las clases del modelo, de manera que cuando (1) es *significativamente* mayor que (2), incluso en el caso en que (1) sea muy pequeño, aceptaremos la relevancia de la clase c o lo que es igual, diremos que la clase c es relevante en el sistema de clasificación. En caso contrario, si (1) no es *significativamente* mayor que (2) para todos los elementos $x \in X$, incluso en el caso en que (1) sea muy grande, consideraremos la clase c como no relevante en el problema que se esta tratando.

Esta aproximación representa la base para un análisis formal de la relevancia, en la búsqueda de la familia más ajustada de clases que contengan las mejores características explicativas.

El problema de la *relevancia* puede, por tanto, ser abordado como un problema de *reducción de dimensionalidad*: una clase que no aporta información adicional acerca de cómo debe ser clasificado un objeto debe ser eliminada. De nuevo, la clave está en determinar cómo y cuáles de esas clases deben ser eliminadas o transformadas manteniendo la capacidad explicativa. En la práctica, buscaremos siempre un número *apropiado* de clases (cuantas menos, en principio, mejor) explicando *al máximo posible* el problema planteado.

Obsérvese que el usuario o decisor puede no tener problema alguno en aceptar que los objetos pertenezcan simultáneamente a varias clases, incluso un fuerte solapamiento *nítido* ($\mu_c(x) = \mu_k(x) = 1$ para algún $c \neq k$), si dicha situación es considerada *relevante* para el verdadero problema de clasificación.

La *relevancia* no debe ser confundida con la *redundancia* que introducimos a continuación. Sus fines son diferentes, las técnicas aplicadas serán diferentes, y cada problema será abordado en un momento diferente del estudio.

3.2 Redundancia

Una vez que la familia inicial de clases ha sido analizada y las clases *no relevantes* se han suprimido, podemos asumir que cada clase aporta algún tipo de información útil. Pero todavía las clases pueden solaparse.

Desde un punto de vista puramente de representación, cuanto menos solapamiento mejor. La *redundancia* se refiere en principio a cierta *ortogonalidad* de la familia de clases, que pasa a jugar el papel de *sistema de representación* del conjunto de objetos.

La *redundancia* sugiere la posible existencia de una representación alternativa, que habrá de ser encontrada redefiniendo algunas de las clases (no hay que olvidar las dificultades que entraña siempre redefinir antiguas clases, o definir nuevas clases). En general, las clases necesitan tener algún significado para el usuario o decisor, de otra manera no sería posible confiar mínimamente en las asignaciones numéricas realizadas sobre los grados de pertenencia.

Una vez finalizado el estudio sobre la *relevancia*, la *redundancia* puede ser estudiada a través de la regla recursiva que juega el papel de conjunción φ. De hecho, el valor

$$\varphi\{\mu_c(x), \mu_k(x)\} \tag{3}$$

puede ser entendido como un grado de solapamiento entre clases $c, k \in C, c \neq k$ con respecto al objeto x. En particular, en el contexto de particiones nítidas,

$$\mu_c(x) \in \{0,1\}, \forall x \in X, \forall c \in C$$

y se asume la existencia de una única clase c_x a la que tal objeto pertenece, de manera que $\mu_{c_x}(x) = 1$ y $\mu_k(x) = 0$ para todo $k \neq c$. Así, para cada $x \in X$

$$\phi_n\{\mu_c(x), \quad c \in C\} = \max\{\mu_c(x) \,/\, c \in C\} = 1,$$

y

$$\varphi_n\{\mu_c(x), \mu_k(x)\} = \min\{\mu_c(x), \mu_k(x)\} = 0, \ \forall c \neq k.$$

Recubrimiento y solapamiento, según han sido introducidos, permiten un desarrollo que tenga en cuenta todos los desarrollos teóricos sobre operadores de agregación y reglas recursivas (ver, entre otros, los trabajos de B. de Baets [2], V. Cutello *et al.* [5], J. Dombi [6, 7], J. C. Fodor *et al.* [8, 9], J. Montero *et al.* [10], y R. R. Yager [15, 16, 17]). El problema de la *redundancia* se va a tratar construyendo una nueva familia de clases a partir de la familia original de forma que se reduzcan los solapamientos sin perder las características de la familia original y con mínima pérdida de información.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto PB98-0825.

Referencias

[1] A. del Amo, J. Montero y E. Molina. Representation of consistent recursive rules. *European Journal of Operational Research* (en prensa).

[2] B. de Baets (1998). Uninorms: the known classes, en D. Ruan, H.A. Abderrahim y P. D'hont (editores) *Fuzzy Logic and Intelligent Technologies for Nuclear Science and Industry* (World Scientific, Singapore), pp. 21–28.

[3] J.C. Bezdek y J.D. Harris. Fuzzy partitions y Relations: an axiomatic basis for clustering, *Fuzzy Sets and Systems* **1**, Pág. 111-127, 1978.

[4] J.C. Bezdek. Fuzzy models for pattern recognition: background, significance and key points en J.C. Bezdek and S.K. Pal (editores) *Fuzzy Models for Pattern Recognition* (IEEE Press, New York), Pág. 1–27, 1992.

[5] V. Cutello y J. Montero. Recursive connective rules, *International Journal of Intelligent Systems*, **14-1**, Pág. 3–20, 1999.

[6] J. Dombi. Basic concepts for a theory of evaluation: the aggregative operator. *European Journal of Opearational Research*, **10**, Pág. 282–293, 1982.

[7] J. Dombi. A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, **8**, Pág. 149–163, 1982.

[8] J.C. Fodor y M. Roubens (1994). *Fuzzy Preference Modelling and Multicriteria Decision Support*, Kluwer Academic Publishers.

[9] J.C. Fodor y M. Roubens. Valued preference structures, *European Journal of Operational Research*, **79**, Pág. 277–286, 1994.

[10] J. Montero, J. Tejada y V. Cutello. A general model for deriving preference structures from data, *European Journal of Operational Research* **98**, Pág. 98–110, 1997.

[11] M. Roubens. Pattern classification problems and fuzzy sets, *Fuzzy Sets and Systems* **1**, Pág. 239–253, 1978.

[12] E.H. Ruspini. A new approach to clustering, *Information and Control* **15**, Pág. 22–32, 1969.

[13] H. Thiele. A Characterization of RUSPINI-Partitions by Similarity Relations, *IPMU'96*, Pág. 389-394, 1996.

[14] H. Thiele. A Characterization of Arbitrary RUSPINI-Partitions by Fuzzy Similarity Relations, *IPMU'96*, Pág. 131-134, 1996.

[15] R.R. Yager (1993). Families of OWA operators, *Fuzzy Sets and Systems* **59**, pp. 125–148.

[16] R.R. Yager (1993). MAM and MOM operators for aggregation, *Information Sciences* **69**, pp. 259–273.

[17] R.R. Yager y A. Rybalov (1996.) Uninorm aggregation operators, *Fuzzy Sets and Systems* **80**, pp. 111–120.

[18] L.A. Zadeh. Fuzzy Sets, *Information and Control* **8**, Pág. 338–353, 1965.

[19] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Transactions on Systems, Man and Cybernetics*, **1**, Pág. 28–44, 1973.

Fuzzy Automata for imperfect string matching

J.R. González de Mendívil, J.R. Garitagoitia, J.J. Astrain, J. Echanobe
Universidad Pública de Navarra, Campus Arrosadía, 31006 Pamplona.
mendivil@unavarra.es, tel: 948 169 093.

Resumen

In many applications it is necessary to determine the similarity of two strings. A widely used notion of string similarity is the Generalized Levenshtein Distance. In this paper, we provide a fuzzy method based on a fuzzy automaton for computing the similarity of two strings and prove that our fuzzy method includes as a particular case the computation of such a distance. The proof is based on comparing the output of the fuzzy automaton with the output value obtained from the computation of the algorithm of Wagner and Fischer (the standard way for computing the Generalized Levenshtein Distance).

Keywords: Fuzzy automata, edition operations, string similarity, edit distance.

1 Introduction

In many applications of pattern recognition a question that has attracted much interest is that of quantifying the similarity of two strings. A review of such distance measures and their applications are in [2, 5, 10, 12]. The most promising of all measures used to compare strings is the one that relates them using various edit operations [12]. The edit operations frequently considered are the deletion, insertion and substitution of individual symbols. Using those edit operations a widely used notion of string distance is the Generalized Levenshtein Distance [6]: the minimum sum of the edit distances associated with the operations required to edit one string to another.

The algorithm of Wagner and Fischer (WF) [14] is usually referred to as the standard solution for computing the Generalized Levenshtein Distance (see section 4). It is based on dynamic programming and has a time complexity of $O(m \times n)$ where m and n are the lengths of the two strings to be compared. Several variants of the string edit distance have been proposed, including the constrain edit distance [9], the normalized edit distance [7] and the parametric edit distance [1]. Another interesting work provides a stochastic interpretation of the string edit distance [11].

In this paper, we introduce a fuzzy method for computing the similarity between two strings. In our method the similarity between two strings α and ω will be the membership value of α in a fuzzy language associated to ω. Such a language is generated by a fuzzy automaton whose transitions models every possible edit operations for matching an arbitrary input string to the target string ω. The individual edit operations (deletion, insertion and substitution of individual symbols) have associated fuzzy values that may be interpreted as the 'cost' to deal with such operations. The fuzzy transitions of the automaton can be calculated using different t-conorms and t-norms. This capability makes the automaton very general; in particular, we prove that the proposed fuzzy automaton behaves as the WF algorithm when using the *maximum* and *product* as the t-conorm and t-norm respectively. The utilization of a t-conorm and a t-norm in the fuzzy automaton for a particular problem comes determined by their suitable behavior in such a problem.

The rest of the paper is organized as follows: In section 2 the fuzzy automaton for modeling the edition operations is introduced. The section 3 is devoted to show the algorithm which computes the transitions of the fuzzy automaton, and its time complexity. In Section 4, it is proved that the proposed fuzzy automaton behaves as the WF algorithm when using the *maximum* and *product* as the t-conorm and t-norm respectively. Finally, conclusions and references end the paper.

2 Fuzzy automata for correcting edition errors

Let Σ be a finite set of symbols (alphabet) and Σ^* be the set of all strings over Σ. Let $\alpha \in \Sigma^*$ and $\omega \in \Sigma^*$, $\alpha = x_1 x_2 \ldots x_m$ and $\omega = a_1 a_2 \ldots a_n$, two arbitrary

Observed string	Operation	Transition	Value	Effect
$x_1 \dots x_{k-1}\ x\ x_{k+1} \dots x_m$	no error c^k_{aa} $(x = a)$	$\mu(q,p,a)$	1	$x_1 \dots x_{k-1}\ a\ x_{k+1} \dots x_m$
$x_1 \dots x_{k-1}\ x\ x_{k+1} \dots x_m$	insertion i^k_a	$\mu(q,p,\varepsilon)$	$\mu_{i^{qp}_a}$	$x_1 \dots x_{k-1}\ a\ x\ x_{k+1} \dots x_m$
$x_1 \dots x_{k-1}\ x\ x_{k+1} \dots x_m$	substitution c^k_{xa} $(x \neq a)$	$\mu(q,p,x)$	$\mu_{c^{qp}_{xa}}$	$x_1 \dots x_{k-1}\ a\ x_{k+1} \dots x_m$
$x_1 \dots x_{k-1}\ x\ x_{k+1} \dots x_m$	deletion d^k_x	$\mu(q,q,x)$	$\mu_{d^q_x}$	$x_1 \dots x_{k-1}\ x_{k+1} \dots x_m$

Figura 1: Interpretation of the fuzzy transitions.

strings. In this section we provide a fuzzy method in order to quantify the similarity between α and ω. In the following α will be called the *observed string*, and ω the *pattern*.

The proposed fuzzy method begins by defining, for $\omega \in \Sigma^*$, a finite deterministic automaton [3] which accepts (recognizes) such string. The automaton $M(\omega) = (Q, \Sigma, \delta, q_0, \{q_n\})$ is defined as follows:

- $Q = \{q_0, q_1, \dots, q_n\}$ is the set of states.
- Σ is the alphabet.
- q_0 and q_n are the start and final states respectively.
- The transition function $\delta : Q \times Q \times \Sigma \longrightarrow \{0,1\}$ is defined as

a) $\delta(q_{k-1}, q_k, a_k) = 1$ $(1 \leq k \leq n)$ where a_k is the k-th symbol in the string ω;
b) All the rest values of $\delta(\cdot)$ are set to 0.

The proposed method must to provide a fuzzy matching between α and ω. For that purpose a fuzzy automaton [8, 13], based on the automaton $M(\omega)$, denoted by $MF(\omega)$ is introduced. This fuzzy automaton models every possible insert, delete or substitution operations when it carries out the matching between the observed string α and the pattern ω.

The fuzzy automaton $MF(\omega) = (Q, \Sigma, \mu, \sigma, \eta)$ is defined as follows:

- Q and Σ are the same sets as in $M(\omega)$.
- $\sigma : Q \longrightarrow [0,1]$ is the fuzzy distribution of start states, where $\sigma(q_0) = 1$ and $\sigma(q_k) = 0$ $(0 < k \leq n)$. So, the fuzzy start state will be $\tilde{\sigma} = \{(q, \sigma(q)) \mid q \in Q\}$[1].
- $\eta : Q \longrightarrow [0,1]$ is the fuzzy distribution of final states, where $\eta(q_n) = 1$ and $\eta(q_k) = 0$ $(0 \leq k < n)$.
- The fuzzy transition function $\mu : Q \times Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow [0,1]$ is defined by the following procedure (ε denotes the empty string):

(i) If $\delta(q,p,a) = 1$, $q, p \in Q$, $a \in \Sigma$, then
  (i.a) $\mu(q,p,a) = 1$ (the original transition)
  (i.b) $\mu(q,p,\varepsilon) \in [0,1]$ (transition for insert op.)
  (i.c) $\forall x \in \Sigma$, $x \neq a$: $\mu(q,p,x) \in [0,1]$ (transition for substitution operation)

(ii) $\forall q \in Q$, $\forall x \in \Sigma$: $\mu(q,q,x) \in [0,1]$ (transition for deletion operation)

(iii) The rest of fuzzy transitions values are set to 0.

Let us now provide an interpretation of the previous fuzzy transition function. Let $\alpha = x_1 \dots x_{k-1}\ x\ x_{k+1} \dots x_m$ be an observed string, and assume that $MF(\omega)$ is going to engage the input symbol x in α:

- (Case i.a) If $\mu(q,p,a)$ is applicable, then there is no error: $x = a$.
- (Case i.b) If $\mu(q,p,\varepsilon)$ is applicable, then the transition simulates the insertion of the symbol[2] a in order to recover a possible deletion of such symbol in the string α.
- (Case i.c) If $\mu(q,p,x)$ $(x \neq a)$ is applicable, then the transition simulates the substitution of the symbol x by a in order to recover a change of a by x in the observed string α.
- (Case ii) If $\mu(q,q,x)$ is applicable, then the transition simulates the deletion of the symbol x in order to correct an insertion of x in the observed string α.

The figure 1 summarizes the previous paragraph. It also introduces part of the notation used in the paper.

Note that the values $\mu_{i^{qp}_a}, \mu_{c^{qp}_{xa}}, \mu_{d^q_x} \in [0,1]$, $\forall x, a \in \Sigma$, $\forall q, p \in Q$, determine the performance of the $MF(\omega)$. The larger/smaller the value of a fuzzy transition are, the more possible/non-possible is the corresponding edit operation to be handled.

The fuzzy automaton has to deal with observed input strings of symbols (not only with individual symbols) and also with fuzzy states. A new fuzzy transition function is defined. Let $\mathcal{F}(Q)$ be the possible fuzzy sets in Q. The function $\hat{\mu} : \mathcal{F}(Q) \times \Sigma \longrightarrow \mathcal{F}(Q)$ is defined as (equation 1):

$$\hat{\mu}(\tilde{P}, x) = \{(p, \mu) \mid \mu = \oplus_{q \in Q}(\mu(q,p,x) \otimes \mu_{\tilde{P}}(q)), p \in Q\}, \text{ with } \tilde{P} \text{ in } \mathcal{F}(Q), x \in \Sigma$$

[1] We will use the following equivalent notations on fuzzy sets $\tilde{A} = \{(x, \mu) \mid x \in U, \mu \in [0,1]\}$ or $A : U \longrightarrow [0,1]$, and for each $x \in U$, $\mu_{\tilde{A}}(x)$ is the membership value μ in the pair (x, μ) of $\tilde{A}$, or equivalently $\mu_{\tilde{A}}(x) = A(x)$.

[2] The symbol a is univocally defined by the pair of states q, p as it is derived from the definition of $M(\omega)$

Input: $M(\omega) = (Q, \Sigma, \delta, q_0, \{q_n\})$, $Q = \{q_0, \ldots, q_n\}$, n **is the length of the chain** ω
the lists $\mu_{d_x^{q_j}}$, $\mu_{c_{xa}^{q_{i-1}q_i}}$ **and** $\mu_{i_a^{q_{i-1}q_i}}$, $\forall a, x \in \Sigma$, $\forall i : 1 \ldots n$, $\forall j : 0 \ldots n$
α **observed string, lenght** m (x_k **k-th symbol of** α)
Output: **the value** $MF(\omega, \alpha)$
where $MF(\omega) = (Q, \Sigma, \mu, \sigma, \eta)$ **is the fuzzy automaton for** $M(\omega)$

algorithm computation
  initialdistrib;
  $\forall k : 1..m$:
    transition(k);
    ε-closure;
  decision
endalgorithm
procedure initialdistrib
  $\forall i : 1..n$:
    $Q(q_i) := 0$;
  $Q(q_0) := 1$;
  ε-closure
endprocedure
procedure decision
  $MF(\omega, \alpha) := Q(q_n)$
endprocedure

procedure transition(k)
  $\forall i : 0..n$:
    $C_1 := Q(q_i) \otimes \mu_{d_x^{q_i}}$;
    $C_2 := Q(q_{i-1}) \otimes \mu_{c_{xa}^{q_{i-1}q_i}}$ ^{a};
      where a is that $\delta(q_{i-1}, q_i, a) = 1$.
    $Q'(q_i) := C_1 \oplus C_2$;
  $\forall i : 0..n$: $Q(q_i) := Q'(q_i)$
endprocedure
procedure ε-closure
  $\forall i : 1..n$:
    $Q(q_i) := \max(Q(q_i), (Q(q_{i-1}) \otimes \mu_{i_a^{q_{i-1}q_i}}))$;
      where a is that $\delta(q_{i-1}, q_i, a) = 1$.
endprocedure

$^a Q(q_{-1}) = 0$.

Figura 2: Algorithm for the fuzzy automaton

where the operators $\oplus$ and $\otimes$ denotes a t-conorm and a t-norm respectively [15, 4].

Finally $\mu^* : \mathcal{F}(Q) \times \Sigma^* \longrightarrow \mathcal{F}(Q)$ is defined as

i) $\mu^*(\tilde{P}, \varepsilon) = \mu^\varepsilon(\tilde{P})$, $\tilde{P}$ in $\mathcal{F}(Q)$

ii) $\mu^*(\tilde{P}, \alpha x) = \mu^\varepsilon(\hat{\mu}(\mu^*(\tilde{P}, \alpha), x))$, $\tilde{P}$ in $\mathcal{F}(Q)$, $x \in \Sigma$, $\alpha \in \Sigma^*$.

The function $\mu^\varepsilon : \mathcal{F}(Q) \longrightarrow \mathcal{F}(Q)$ can be interpreted as a fuzzy version of the classical closure by empty string (ε-CLOSURE) for non-deterministic automata [3]. The procedure to calculate $\mu^\varepsilon(\tilde{P})$, with $\tilde{P}$ in $\mathcal{F}(Q)$, is:

• Let $\langle q_{k_0} q_{k_1} \ldots q_{k_l} \rangle$ be an arbitrary finite ($l > 0$) sequence of states of Q such that $\forall i, j$, $0 \leq i, j \leq l$, $i \neq j$ then $q_{k_i} \neq q_{k_j}$.
• Let $\tilde{E}(q)$ be the fuzzy set in Q representing the reachable set of states from q by repeatedly using transitions by empty string, that is,

$\tilde{E}(q) = \{(p, \mu) \mid \mu = \oplus_{\langle q_{k_0} \ldots q_{k_l} \rangle, q_{k_0} = q, q_{k_l} = p} (\otimes_{0 < i \leq l} \mu(q_{k_{i-1}}, q_{k_i}, \varepsilon)), p \in Q\}$

Finally (equation 2),

$\mu^\varepsilon(\tilde{P}) = \{(p, \mu) \mid \mu = \mu_{\tilde{P}}(p) \oplus (\oplus_{q \in Q} (\mu_{\tilde{E}(q)}(p) \otimes \mu_{\tilde{P}}(q))), p \in Q\}$, with $\tilde{P}$ in $\mathcal{F}(Q)$

Thus, the defined automaton, has transitions by empty strings and the calculation of such transitions is given by equation 2. Note however that, as ε has null length ($|\varepsilon| = 0$) it is verified $\varepsilon \equiv \varepsilon\varepsilon \equiv \varepsilon\varepsilon\varepsilon \ldots$. This forces the calculation of $\mu^\varepsilon(\tilde{P})$, for a fuzzy state $\tilde{P}$, to be idempotent; that is, $\mu^\varepsilon(\mu^\varepsilon(\tilde{P})) = \mu^\varepsilon(\tilde{P})$. This circumstance limits the possible t-conorms to be used. The only t-conorm that preserves the idempotency in equation 2 is the *maximum* operator. In consequence, for the computation of $\mu^\varepsilon(\tilde{P})$ in the equation 2, we have to use as t-conorm the *maximum* operator, for the rest of equations we use any t-conorm and t-norm [15, 4].

We define the fuzzy language generated by $MF(\omega)$ as the fuzzy set in Σ^*:

$\tilde{L}(MF(\omega)) = \{(\alpha, \mu) \mid \mu = \oplus_{q \in Q} (\mu_{\mu^*(\tilde{\sigma}, \alpha)}(q) \otimes \eta(q)), \alpha \in \Sigma^*\}$

As $M(\omega)$ has a unique final state q_n, then for an observed string α the fuzzy automaton $MF(\omega)$ calculates the membership value $\mu_{\mu^*(\tilde{\sigma}, \alpha)}(q_n)$ (provided that $\eta(q_n) = 1$ and $\eta(q_k) = 0$ with $0 \leq k < n$) in the fuzzy language $\tilde{L}(MF(\omega))$. Therefore, the fuzzy automaton may be interpreted as a *similarity operator* such that for each observed input string α in Σ^* it assignees the value

$MF(\omega, \alpha) = \mu_{\mu^*(\tilde{\sigma}, \alpha)}(q_n) \in [0, 1], \omega, \alpha \in \Sigma^*$

In the following section we provide an algorithm for the computation of the fuzzy automaton.

3 Algorithm for the fuzzy automaton

The main program (figure 2) is given by the algorithm **computation**. This algorithm starts by building the initial distribution of states which is given by σ (from the definition of $MF(\omega)$). The nucleus of this algorithm is a loop in which the procedures **transition**

	$\mu(q_0)$	$\mu(q_1)$	$\mu(q_2)$	$\mu(q_3)$
$\tilde{\sigma}$	1	0	0	0
$\tilde{Q}_0 \equiv \mu^\varepsilon(\tilde{\sigma})$	$1\ \surd$	0 $\mu_{i_a}\ \surd$	0 $\mu_{i_a}\mu_{i_e}\ \surd$	0 $\mu_{i_a}\mu_{i_e}\mu_{i_e}\ \surd$
$\hat{\mu}(\mu^\varepsilon(\tilde{\sigma}), a)$	$\mu_{d_a}\ \surd$	$\mu_{i_a}\mu_{d_a}$ $\mu_{c_{aa}}\ \surd$	$\mu_{i_a}\mu_{i_e}\mu_{d_a}$ $\mu_{i_a}\mu_{c_{ae}}\ \surd$	$\mu_{i_a}\mu_{i_e}\mu_{i_e}\mu_{d_a}$ $\mu_{i_a}\mu_{i_e}\mu_{c_{ae}}\ \surd$
$\tilde{Q}_1 \equiv \mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\tilde{\sigma}), a))$	$\mu_{d_a}\ \surd$	$\mu_{c_{aa}}\ \surd$ $\mu_{d_a}\mu_{i_a}$	$\mu_{i_a}\mu_{c_{ae}}$ $\mu_{c_{aa}}\mu_{i_e}\ \surd$	$\mu_{i_a}\mu_{i_e}\mu_{c_{ae}}$ $\mu_{c_{aa}}\mu_{i_e}\mu_{i_e}\ \surd$
$\hat{\mu}(\mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\tilde{\sigma}), a)), b)$	$\mu_{d_a}\mu_{d_b}\ \surd$	$\mu_{c_{aa}}\mu_{d_b}\ \surd$ $\mu_{d_a}\mu_{c_{bq}}$	$\mu_{c_{aa}}\mu_{i_e}\mu_{d_b}$ $\mu_{c_{aa}}\mu_{c_{be}}\ \surd$	$\mu_{c_{aa}}\mu_{i_e}\mu_{i_e}\mu_{d_b}$ $\mu_{c_{aa}}\mu_{i_e}\mu_{c_{be}}\ \surd$
$\tilde{Q}_2 \equiv \mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\tilde{\sigma}), a)), b))$	$\mu_{d_a}\mu_{d_b}\ \surd$	$\mu_{c_{aa}}\mu_{d_b}\ \surd$ $\mu_{d_a}\mu_{d_b}\mu_{i_a}$	$\mu_{c_{aa}}\mu_{c_{be}}\ \surd$ $\mu_{c_{aa}}\mu_{d_b}\mu_{i_e}$	$\mu_{c_{aa}}\mu_{i_e}\mu_{c_{be}}\ \surd$ $\mu_{c_{aa}}\mu_{c_{be}}\mu_{i_e}$

Figura 3: The behavior of the fuzzy automaton $MF(aee)$ for the observed string $\alpha \equiv ab$. $\tilde{Q}_i$ $(i = 0, 1, 2)$ is the fuzzy state reached in the i-th step. This fuzzy state is composed by the states in the heading of the columns (q_0, q_1, q_2, q_3) together with the selected values $(\surd)$ appearing in the i row.

and ε-`closure` are executed for every fuzzy symbol x_k from the observed string α. Such procedures compute respectively the equations 1 and 2. Finally, the procedure `decision`, computes the final distribution of states given by η. Due to the linearity of $M(\omega)$ it is possible to make the following simplifications. In equation 1, we can see that there are only two possible transitions to reach the state q_i: One from the state q_{i-1}, representing either a substitution operation or a non-error operation (given by C_2 in figure 2) and the other one from the same state q_i representing a delete operation (given by C_1 in figure 2). Also, in the equation 2, it can be proved that the state q_i is only affected by the value of the state q_{i-1} and the transition representing the insert operation between q_{i-1} and q_i.

Given an automaton $M(\omega)$, the complexity of the procedure `transition` is given by $\mathcal{O}(n)$, where n is the length of the string ω. In the other hand, the procedure ε-`closure` has a complexity $\mathcal{O}(n)$. Therefore, the complete execution of the algorithm `computation` for an observed string α of length m is $\mathcal{O}(m \times n)$.

In the following, an example illustrating the behavior of the algorithm for the fuzzy automaton, is showed. Let $\omega \equiv aee$ be the pattern string obtained from the alphabet $\Sigma = \{a, b, e\}$. The finite deterministic automaton $M(\omega)$ which accepts the string ω and the associated fuzzy automaton $MF(\omega)$ are calculated as it was explained in the section 2. The fuzzy automaton has been obtained from its definition in the previous section. The fuzzy automaton deals with strings from Σ^* which differ from ω in a certain number (zero included) of insert, delete and substitution edit operations. For simplicity, the values $\mu_{i_a^{qp}}, \mu_{c_{xa}^{qp}}, \mu_{d_x^q} \in [0, 1]$ are assumed to be independent of the states.

Given the observed string $\alpha \equiv ab$, the computation of the fuzzy automaton is given by $MF(\omega, \alpha) = \mu_{\mu^*(\tilde{\sigma},\alpha)}(q_3)$, where $\mu^*(\tilde{\sigma}, \alpha) = \mu^*(\tilde{\sigma}, ab) = \mu^\varepsilon(\hat{\mu}(\mu^*(\tilde{\sigma}, a), b)) = \mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\hat{\mu}(\mu^*(\tilde{\sigma}, \varepsilon), a)), b)) = \mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\hat{\mu}(\mu^\varepsilon(\tilde{\sigma}), a)), b))$

The figure 3 shows the behavior of the fuzzy automaton $MF(aee)$ when it computes the input string ab. It also includes the intermediate steps followed by the automaton. The $\otimes$ operator used in this example is the *algebraic product* and the $\oplus$ operator is the *maximum*. The values $\mu_{i_\cdot}$, $\mu_{c_{\cdot\cdot}}$, $\mu_{d_\cdot}$ are restricted to be $1 > \mu_{i_\cdot} > \mu_{c_{\cdot\cdot}} > \mu_{d_\cdot} > 0$. Moreover, we assume that the product of any pair of such values is always less than any of the three values. The marked $(\surd)$ values represent the maximum values after the equations 1 or 2 have been computed for every possible case.

When the computation of the algorithm for the fuzzy automaton finishes, the value assigned to $\alpha \equiv ab$ is $MF(\omega, \alpha) = \mu_{c_{aa}}\mu_{i_e}\mu_{c_{be}}$. In other words, the fuzzy automaton recognizes the observed string α as the pattern string $\omega \equiv aee$ with a similarity value of $\mu_{c_{aa}}\mu_{i_e}\mu_{c_{be}}$. By following the trace of the computation, it can be observed how the value $\mu_{c_{aa}}\mu_{i_e}\mu_{c_{be}}$ may be interpreted as the better possibility for matching the strings $\alpha \equiv ab$ and $\omega \equiv aee$ by doing the substitution of a by itself $(\mu_{c_{aa}} = 1)$, the insertion of symbol e (μ_{i_e}) and the substitution of b by e $(\mu_{c_{be}})$.

4 Fuzzy automaton computes edit distances

In this section, we prove that the proposed fuzzy automaton behaves as the WF algorithm when the *maximum* and *product* are used as the t-conorm and t-norm respectively.

The algorithm of the Wagner and Fischer (WF) [14] is usually referred to as the standard solution to the

Input: C_{i_a}, $C_{c_{xa}}$, C_{d_x}, $x, a \in \Sigma$
$\alpha = x_1 x_2 \ldots x_m \in \Sigma^*$, $\omega = a_1 a_2 \ldots a_n \in \Sigma^*$
Output: $d(\alpha, \omega)$
Procedure:

```
D(0,0):= 0;
for i:= 1 to m do D(i,0):= D(i-1,0) + C_{d_{x_i}};
for j:= 1 to n do D(0,j):= D(0,j-1) + C_{i_{a_j}};
for i:= 1 to m do
    for j:= 1 to n do
        m1:= D(i-1,j-1) + C_{c_{x_i a_j}}; /* substitution */
        m2:= D(i-1,j) + C_{d_{x_i}}; /* deletion */
        m3:= D(i,j-1) + C_{i_{a_j}}; /* insertion */
        D(i,j):= min(m1,m2,m3);
d(α,ω):= D(m,n)
```

Figura 4: Wagner and Fischer Algorithm

problem of computation the minimum edition distance between two strings. We introduce the string distance and give the basic algorithm for this computation.

Let Σ be an alphabet of symbols, $\alpha = x_1 x_2 \ldots x_m \in \Sigma^*$, and $\omega = a_1 a_2 \ldots a_n \in \Sigma^*$, with $m, n \geq 0$. The elementary edit operations considered to transform α into ω are:

(a) Substitution of a symbol $x \in \Sigma$, in α, by a symbol $a \in \Sigma$, in ω, and $x \neq a$ (c_{xa} operator).
(b) Insertion of a symbol $a \in \Sigma$, in ω (i_a operator).
(c) Deletion of a symbol $x \in \Sigma$, in α (d_x operator).

Depending on the particular application, certain edit operations, may be more likely that others. In order to take into account this observation, it makes sense to introduce 'cost' of edit operations. We denote $C_{c_{xa}}$ ($C_{c_{xa}} = 0$, $x = a$), C_{i_a}, C_{d_x}, with $x, a \in \Sigma$, for the costs of substitution, insertion and deletion operators respectively. The costs of any edit operation are assumed to be a positive real number.

Let $\pi = \theta_1 \ldots \theta_r$ be a sequence of edit operations for transforming a string α into another string ω, the cost associated to this sequence π is $C(\pi) = \sum_{i=1}^{r} C_{\theta_i}$. The edit distance between α and ω is defined by $d(\alpha, \omega) = \min\{C(\pi) |\ \pi$ is a sequence of edit operations which transform α into $\omega\}$.

The WF algorithm calculates $d(\alpha, \omega)$ according to the previous definition. Basically, it is a dynamic programming procedure (i.e. a particular breath-first search). The algorithm (figure 4) computes the elements of the $(m+1) \times (n+1)$ matrix $D(i,j)$, row by row from the left to right. In each element $D(i,j)$, the minimum accumulative costs are stored for transforming the substrings $\alpha_i = x_1 x_2 \ldots x_i$ into $\omega_j = a_1 a_2 \ldots a_j$. The algorithm ends when $D(m,n)$ is computed. This value is the Generalized Levenshtein Distance for the strings α and ω.

In order to make the proof, we assume that the fuzzy values μ are independent of the state and the following relations hold (relations 1): $\forall x, a \in \Sigma$

$$C_{c_{xa}} = -\log(\mu_{c_{xa}}), C_{i_a} = -\log(\mu_{i_a}), C_{d_x} = -\log(\mu_{d_x})$$

We prove that the value $d(\alpha, \omega)$ computed by the WF algorithm is equal to $-\log(MF(\omega, \alpha))$, where $MF(\omega, \alpha)$ is the value computed by the fuzzy automaton and $\alpha, \omega \in \Sigma^*$. In the following lemma, $\tilde{Q}_0$ denotes the fuzzy state obtained in the initial distribution (`initialdistrib`), and $\tilde{Q}_i$ ($1 \leq i \leq m$) denotes the fuzzy state obtained in i-th complete transition (`transition(`i`)` and ε-`closure`) of the fuzzy automaton.

Lemma: Let $\alpha = x_1 \ldots x_m$, $\alpha \in \Sigma^*$, and $\omega = a_1 \ldots a_n$, $\omega \in \Sigma^*$ be two arbitrary strings. If relations 1 holds then: $\forall i : 0 \ldots m, \forall j : 0 \ldots n :\ D(i,j) = -\log(Q_i(q_j))$.

Proof: By induction over $i : 0 \ldots m$.

(A) *Basis*: $\forall j : 0 \ldots n :\ D(0,j) = -\log(Q_0(q_j))$

By induction over $j : 0 \ldots n$.

(A1) *Basis*: $D(0,0) = -\log(Q_0(q_0))$. It holds because $D(0,0) = 0$ and $Q_0(q_0) = \sigma(q_0) = 1$.

(A2) *Hypothesis*: $D(0, j-1) = -\log(Q_0(q_{j-1}))$.

(A3) *Step*: $D(0,j) = D(0,j-1) + C_{i_{a_j}}$ and $Q_0(q_j) = \max(\sigma(q_j), Q_0(q_{j-1}) \cdot \mu_{i_{a_j}})$. As $\sigma(q_j) = 0$ then $Q_0(q_j) = Q_0(q_{j-1}) \cdot \mu_{i_{a_j}}$, by using (A2) and the relations 1 then $-\log(Q_0(q_j)) = -\log(Q_0(q_{j-1}) \cdot \mu_{i_{a_j}}) = -\log(Q_0(q_{j-1})) - \log(\mu_{i_{a_j}}) = D(0, j-1) + C_{i_{a_j}} = D(0,j)$.

(B) *Hypothesis*: $\forall j : 0 \ldots n :\ D(i-1, j) = -\log(Q_{i-1}(q_j))$

(C) *Step*: It will be proved that $\forall j : 0 \ldots n :\ D(i,j) = -\log(Q_i(q_j))$ By induction over $j : 0 \ldots n$.

(C1) *Basis*: $D(i,0) = -\log(Q_i(q_0))$. It holds because $D(i,0) = D(i-1,0) + C_{d_{x_i}}$ and for the fuzzy automaton $Q'_i(q_0) = \max(Q_{i-1}(q_0) \cdot \mu_{d_{x_i}}, Q_{i-1}(q_{-1}) \cdot \mu_{c_{x_i a_0}}) = Q_{i-1}(q_0) \cdot \mu_{d_{x_i}}$, due to $Q_{i-1}(q_{-1}) = 0$ by definition. Therefore, $Q_i(q_0) = Q'_i(q_0) = Q_{i-1}(q_0) \cdot \mu_{d_{x_i}}$ because ε-`closure` does not change this value. Finally, using (B) and the relations 1, $-\log(Q_i(q_0)) = -\log(Q_{i-1}(q_0) \cdot \mu_{d_{x_i}}) = -\log(Q_{i-1}(q_0)) - \log(\mu_{d_{x_i}}) = D(i-1,0) + C_{d_{x_i}} = D(i,0)$.

(C2) *Hypothesis*: $D(i,j-1) = -\log(Q_i(q_{j-1}))$.

(C3) *Step*: The WF algorithm computes the values $m_1 = D(i-1,j-1) + C_{c_{x_i a_j}}$, $m_2 = D(i-1,j) + C_{d_{x_i}}$, $m_3 = D(i,j-1) + C_{i_{a_j}}$, and finally $D(i,j) = \min(m_1, m_2, m_3)$.
The fuzzy automaton computes the values $C_1 = Q_{i-1}(q_j) \cdot \mu_{d_{x_i}}$ and $C_2 = Q_{i-1}(q_{j-1}) \cdot \mu_{c_{x_i a_j}}$ then $Q'(q_j) = \max(C_1, C_2)$, and $Q_i(q_j) = \max(Q'(q_j), Q_i(q_{j-1}) \cdot \mu_{i_{a_j}})$.
So, $Q_i(q_j) = \max(Q_{i-1}(q_j) \cdot \mu_{d_{x_i}}, Q_{i-1}(q_{j-1}) \cdot \mu_{c_{x_i a_j}}, Q_i(q_{j-1}) \cdot \mu_{i_{a_j}})$. By using (B), (C2) and the relations 1, it is very simple to prove that $D(i,j) = -\log(Q_i(q_j))$. ■

As $d(\alpha,\omega) = D(m,n)$ and $MF(\omega,\alpha) = Q_m(q_n)$, by using the lemma and the relations 1, it is verified $d(\alpha,\omega) = -\log(MF(\omega,\alpha))$. Therefore, the proposed fuzzy automaton can be also used to compute the Generalized Levenshtein Distance.

5 Conclusions

In this paper, a family of measures of the similarity of two strings are introduced. The proposed method is based on the development of a fuzzy automaton which models the typical edit operations which are used to transform an observed strings into a pattern string. The similarity value is the membership value of the observed string into the fuzzy language associated to the pattern string via the fuzzy automaton.

The fuzzy method has not limit in the number of edit operations to be used in the matching between both strings. In addition, it is provided an algorithm to compute the fuzzy automaton (and by product the fuzzy similarity) which has a time complexity of $O(m \times n)$ where m and n are the lengths of the observed and pattern strings respectively.

The proposed fuzzy automaton is able to compute the Generalized Levenshtein Distance when it uses for the computation of its transitions the *maximum* and the *product* as t-conorm and t-norm respectively. The fuzzy automaton can provide another edit distances by varying the t-conorm and t-norm. For practical applications, it is required further study in order to select the t-conorm and t-norm which best fit to the problem.

Referencias

[1] H. Bunke, J. Csirik; 'Parametric String Edit Distance and its Application to Pattern Recognition'; *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, n. 1, pp. 202-206, Jan. 1995.

[2] P.A.V. Hall, G. R. Dowling; 'Approximate String Matching'; *ACM Computing Surveys*, vol. 12, n. 4, pp. 381-402; Dec. 1980.

[3] J. Hopcroft, J. Ullman; *Introduction to Automata Theory, Languages and Computation*; Addison-Wesley Publishing Company, Reading Massachusetts; 1979.

[4] G.J. Klir, B. Yuan; *Fuzzy sets and fuzzy logic: Theory and applications*; Prentice Hall, N.J., 1995.

[5] K. Kukich; 'Techniques for Automatically Correcting Words in Text'; *ACM Computing Surveys*, vol. 24, n. 4, pp. 377-439; Dec. 1992.

[6] V.I. Levenshtein; 'Binary codes capable of correcting deletions, insertions, and reversals'; *Soviet Physics Docklady*, vol. 10, n. 8; pp. 707-710; 1966.

[7] A. Marzal, E. Vidal; 'Computation of Normalized Edit Distance and Applications'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, n. 9, pp. 926-932, Sep. 1993.

[8] M. Mizumoto, J. Toyoda, K. Tanaka; 'Fuzzy Languages'; *Systems Computers Control*, vol. 1, n. 3; pp. 36-43; 1970.

[9] B. Oommen; 'Constrained String Editing'. *Information Sciences*, vol. 40, pp. 267-284, 1986.

[10] J.L. Peterson; 'Computer programs for detecting and correcting spelling errors'; *Communications of the ACM*, vol. 23, pp. 676-687, 1980.

[11] E.S. Ristad, P.N. Yianilos; 'Learning String-Edit Distance'; *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, n. 5, pp. 522-531, May 1998.

[12] D. Sankoff, J.B. Kruskal; *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*; Addison-Wesley, Reading, Mass. 1983.

[13] M.G. Thomason; 'Finite fuzzy automata, regular fuzzy languages and pattern recognition'; *Pattern Recognition*, vol. 5; pp. 383-390; 1973.

[14] R.A. Wagner, M.J. Fischer; 'The string-to-string correction problem'; *Journal of ACM*, vol. 21, n. 1; pp. 168-173; Jan. 1974.

[15] H.J. Zimmermann; *Fuzzy Set Theory and its Applications*; Kluwer Academic Pub., Boston, MA; 1990.

MORFOLOGÍA MATEMÁTICA BORROSA Y MORFOLOGÍA MATEMÁTICA DE FUNCIONES GREY-LEVEL

P. Burillo
Dpto de Automática y Computación. Universidad Pública de Navarra Campus de Arrosadía 31006. Pamplona. Navarra. E-mail: pburillo @unavarra.es

N. Frago
Dpto de Matemática e Informática. Universidad Pública de Navarra Campus de Arrosadía 31006. Pamplona. Navarra. E-mail: noe.frago @unavarra.es,

R. Fuentes
Dpto de Automática y Computación. Universidad Pública de Navarra Campus de Arrosadía 31006. Pamplona. Navarra. E-mail: rfuentes@unavarra.es

Resumen

En esta comunicación se hace un estudio comparativo entre los operadores erosión y dilatación de la Morfología Matemática de funciones "grey-level " y los operadores erosión y dilatación de la Morfología Matemática borrosa construidas a partir los operadores de grado de inclusión con las condiciones impuestas por Baets-Kerre para las implicaciones borrosas utilizadas.

Palabras claves: Grado de Inclusión, Operador Implicación, Erosión, Dilatación, Morfología Matemática borrosa.

1. INTRODUCCIÓN

La Morfología Matemática, introducida por Matheron en 1964 [4] se ha manifestado como una potente herramienta para el procesamiento digital de imágenes. En ella, si consideramos dos imágenes A y B como subconjuntos ordinarios del plano R^2 o Z^2, en este trabajo se define

- Erosión de A por B como el subconjunto $\xi(A, B)$ dado por

$$\xi(A, B) = \{z \in R^2 \mid B_Z \subset A\} = \bigcap_{b \in B} A_{-b} \quad (1)$$

- Dilatación de A por B como el subconjunto $\mathcal{D}(A, B)$ dado por

$$\mathcal{D}(A, B) = \bigcup_{b \in B} A_{-b} \quad (2)$$

en donde $Bz = \{z + b \mid b \in B\}$ es el subconjunto trasladado de B por z y análogamente A_{-b} y A_b son los trasladados de A por $-b$ y b respectivamente. La imagen B, llamada elemento estructurante, es considerada en este trabajo como un subconjunto centrado en el origen.

Para imágenes con distintos tonos de gris, representadas por funciones $f: D_f \longrightarrow [0, 1]$, $g: D_g \longrightarrow [0, 1]$, siendo D_f y D_g subconjuntos ordinarios de R^2 ó Z^2 se define

- Erosión de f por g como el subconjunto $\xi_{g\text{-}l}(f, g)$ dado por

$$\xi_{g\text{-}l}(f, g)(z) = \inf_{x \in D_g, z+x \in D_f} \{f(z+x) - g(x)\} \quad (3)$$

para $z \in D_f \quad D_g$.

- Dilatación de f por g como el subconjunto $\mathcal{D}_{g\text{-}l}(f, g)$ dado por

$$\mathcal{D}_{g\text{-}l}(f, g)(z) = \sup_{x \in D_g, z-x \in D_f} \{f(z-x) + g(x)\} \quad (4)$$

para $z \in D_f \quad D_g$.

Posteriormente, y en el marco de la teoría de subconjuntos borrosos, se han propuesto generalizaciones de las nociones de erosión y dilatación de imágenes (subconjuntos) ordinarios. Así, si consideramos un operador de implicación borrosa $I: [0, 1]x[0, 1] \longrightarrow [0, 1]$ verificando:

- $I(0, 0) = I(1, 1) = 1, I(1, 0) = 0$
- I decreciente en la 1ª componente
- I creciente en la 2ª componente

resulta evidente que $I(0, b) = 1 \ \forall b \in [0, 1]$ y en particular $I(0, 1) = 1$, luego I generaliza las implicaciones Booleanas. En estas condiciones, es sabido que si n es una negación e I satisface además:

- $I(1, y) = y \ \forall y \in [0, 1]$
- $I(x, y) = I(n(y), n(x)) \ \forall x, y \in [0, 1]$

Entonces $n(x) = I(x, 0) \ \forall x \in [0, 1]$ y $n(n(x)) = x \ \forall x \in [0, 1]$, n es una negación involutiva. Si además se verifica que $I(x, I(y, z)) = I(y, I(x, z)) \ \forall x, y, z \in [0, 1]$ la aplicación $C: [0, 1]^2 \longrightarrow [0, 1]$ dada por $C(x, y) = n(I(x, n(y)))$ con $n(x) = I(x, 0)$ es una t-norma [1, 2,5, 10-14].

Pues bien, si consideramos dos conjuntos (imágenes) borrosos de

- $A: D_A \longrightarrow [0, 1]$
- $B: D_B \longrightarrow [0, 1]$

con dominios de definición D_A y D_B subconjuntos ordinarios de R^2. Llamaremos [6-9, 2, 4]

- Erosión de A por B como el subconjunto $\xi(A, B)$ dado por

$$\xi(A, B)(z) = \inf_{x \in D_A, x-z \in D_B} \{I(B(x-z), A(x))\} \quad (5)$$

- Dilatación de A por B como el subconjunto $\mathcal{D}(A, B)$ dado por

$$\mathcal{D}(A, B)(z) = \sup_{x \in D_A, x-z \in D_B} \{C(B(x-z), A(x))\} \quad (6)$$

Resulta en definitiva, que para dos imágenes con distintos tonos de gris (subconjuntos borrosos) A y B podemos efectuar operaciones y erosiones utilizando las

técnicas difusas o las correspondientes a funciones. El objetivo de este trabajo es analizar ambas posibilidades y compararlas.

2. EROSIONES

Hacemos notar, en primer lugar, que un sencillo cambio de variable permite escribir (7) en la forma

$$\xi(A, B)(z) = \inf_{x\in D_B, x+z\in D_A} \{I(B(x), A(z + x))\} \quad (9)$$

resultando de (7) y (9) que el dominio de definición de $\xi(A, B)$ es $D_A \quad D_B$.

Veamos en primer lugar que el algoritmo de calculo de $\xi(A, B)$ se puede simplificar al conjunto $SopB\cap(D_A\text{-}z)$ con $SopB = \{x \mid B(x) > 0\}$.

Proposición 1.- Para todo z perteneciente a D_A,

$$\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{I(B(x), A(z + x))\}$$

Demostración.- En efecto, si x no pertenece a Sop B entonces $B(x) = 0$ luego $I(B(x), A(z + x)) = I(0, A(z + x)) = 1$.

Algunos casos particulares interesantes se recogen en la siguiente proposición.

Proposición 2.- Para una implicación borrosa I que verifica las propiedades antes mencionadas, se verifica:

1) La expresión

$$\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{A(x + z))\}$$

es cierta para cualquiera A borroso, B ordinario de R^2 ó Z^2 si y sólo si $I(1, b) = 1 \ \forall b\in[0, 1]$.

2) Si A es un subconjunto borroso y B un subconjunto borroso tal que $B(x) = K\in[0, 1]$, $\forall x\in D_B$, entonces

$$\xi(A, B)(z) = I(K, \inf_{x\in SopB\cap(D_A - z)} \{A(x + z)\}) \quad y$$

$$n(k) \le \xi(A, B)(z) \le I(K, \sup_{h\in D_A} A(h))] \ \forall z\in D_{\xi(A,B)}.$$

3) Si A y B son subconjuntos ordinarios la erosión dada por (7) coincide con la erosión ordinaria (1)

Demostración.

1) Como $B(x) = 1 \ \forall x\in$sop B y además $I(1, b) = b$ $\forall b\in[0, 1]$, tenemos: $\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{I(1,$ $A(x + z))\} = \inf_{x\in SopB\cap(D_A - z)} \{A(x + z)\} \ \forall z\in D_A$

Recíprocamente si para todo subconjunto borroso A y para todo subconjunto ordinario B se verifica que $\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{I(1, A(x + z))\}$, en particular se verificara para todo subconjunto borroso A y para el subconjunto ordinario B tal que Sop B = {0} con B(0) = 1. En este caso se cumple que $\xi(A, B)(z) = I(1, A(x + z)) = A(z)$, y como esto se verifica para todo subconjunto borroso A y para cualquier $z\in D_A$, podemos afirmar que $I(1, b) = b \ \forall b\in[0, 1]$.

2) Si $B(x) = K \ \forall x\in$Sop B, entonces tenemos $\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{I(K, A(x+z))\}$ y por ser I creciente para la segunda variable nos queda $\xi(A, B)(z) = I(K, \inf_{x\in SopB\cap(D_A - z)} \{A(x+z)\})$.

$$n(k) = I(k, 0) \le \xi(A, B)(z) \le I(K, \sup_{h\in D_A} A(h))] \ \forall z\in D_{\xi(A,B)}.$$

Propiedad que en términos de imágenes nos dice que la erosión de una imagen A por un conjunto B de un tono de gris k proporciona una imagen cuyos tonos de gris superan el tono complementario n(k). Debemos hacer notar que algunas aplicaciones (Aphelion) de procesamiento de imágenes normalizan este tipo de imágenes a un intervalo limitado por el negro (0) y el blanco (255) ({0, 1, 2, ..., 255}) ocultando este aspecto.

3) Si A y B subconjuntos ordinarios, podemos identificar $A = D_A$ y $Sop\ B = D_B = B$, entonces $\forall z\in D_f \quad D_g$:

$$\xi(A, B)(z) = \inf_{x\in SopB\cap(D_A - z)} \{A(x + z)\} = 1 \Leftrightarrow$$

$\forall x\in$Sop $B\cap D_A$-z, $A(x + z) = 1 \Leftrightarrow \forall x\in$Sop $B\cap D_A$-z, $x+z\in A \Leftrightarrow \forall x\in$Sop $B\cap D_A$-z, $x+z\in(D_B + z)$ y $x+z\in D_A \Leftrightarrow$ $(D_B + z)\subset D_A \Leftrightarrow B+z\subset A$

Verificándose que las erosiones borrosas recuperan la erosión binaria.

Para dos subconjuntos borrosos A y B, considerados como funciones de sus dominios D_A y D_B en [0, 1], aplicamos ahora las expresiones de la erosión de funciones y compararemos los resultados con los que anteceden. Veremos que existen coincidencias y diferencias incluso limitándonos a utilizar elementos estructurantes centrados en el origen y simétricos ($B(x) = B(-x) \ \forall x\in D_B$).

Así, para dos subconjuntos borrosos A y B resulta que

$$\xi_{g\text{-}1}(A, B)(z) = \inf_{x\in D_B\cap D_A - z} \{A(z + x)\text{-}B(x)\} \quad (10)$$

Notemos en primer lugar que a diferencia de la proposición 1, no es posible en general restringir en (10) el recorrido x al conjunto Sop $B \cap D_A$ - z .

Ejemplo 1.- Si consideramos los conjuntos difusos $A(x_0) = 1$ y $A(x) = 0$ si $x \ne x_0$, con $D_A = R^2$

$B(0) = 0.5$ y $B(x) = 0$ si $x \ne 0$, $Sop\ B = \{(0, 0)\}$ y $D_B = \{(-1, -1), (0, 0), (1, 1)\}$. Entonces

$\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in D_B \cap D_A - z} \{A(z + x)\text{-}B(x)\}$ no tiene por que coincidir con $\inf_{x \in SopB \cap D_A - z} \{A(z + x)\text{-}B(x)\}$ como podemos comprobar con los subconjuntos dados, en efecto

$\xi_{g\text{-}I}(A, B)(x_0) = \inf_{x \in D_B \cap D_A - x_0} \{A(x_0 + x)\text{-}B(x)\} = 0$ y

$\xi_{g\text{-}I}(A, B)(x_0) = \inf_{x \in SopB \cap D_A - x_0} \{A(x_0 + x)\text{-}B(x)\} = 0.5$

Si A, B son subconjuntos borrosos con dominio D_A y D_B respectivamente, tales que $B(x) = K \in [0, 1]$ $\forall x \in$ Sop B, entonces: $\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in D_B \cap D_A - z} \{A(z + x)\} - K$

En particular si B es un subconjunto ordinario, entonces

$$\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in D_B \cap D_A - z} \{A(z+x)\} - 1$$

Proposición 3.-

1) Si A, B son subconjuntos borrosos con dominio D_A y D_B respectivamente, tales que $B(x) = K$ $\forall x \in$ Sop B y $A(x) \leq K$ $\forall x \in$ Sop A, $0 < K \leq 1$, entonces:

$$\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in SopB \cap D_A - z} \{A(z + x)\} - K$$

2) Existen subconjuntos borrosos A y B para los que la erosión para funciones $\xi_{g\text{-}I}$ (A, B)(z) y la erosión borrosa no coincide ξ (A, B)(z).

3) Si A y B son subconjuntos ordinarios de U, se verifica:

$\xi_{g\text{-}I}(A, B)(z) = 1$ si y sólo si $B+z \subseteq A$

$\xi_{g\text{-}I}(A, B)(z) = 0$ si y sólo si $B+z \not\subset A$

$\forall z \in D_f$ D_g, es decir, se recupera la erosión binaria.

Demostración

1) Si A es una aplicación de D_A en [0, K] y B una aplicación de D_B en {0, K}, $0 < K \leq 1$ entonces

$\forall x \in$ sop B se verifica $A(z + x)\text{-}B(x) = A(z + x) - K \leq 0$

$\forall x \notin$ sop B se verifica $A(z + x)\text{-}B(x) = A(z + x) - 0 \geq 0$

Luego $\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in SopB \cap D_A - z} \{A(z + x)\text{-}K\}$

$= \inf_{x \in SopB \cap D_A - z} \{A(z + x)\} - K.$

2) En efecto, sean A tal que $A(x_0) = 0$, $A(x_1) = 0.2$, $A(x_2) = 0.4$, $A(x_3) = 0.6$, $A(x_4) = 0.8$, $A(x_5) = 0.9$, $A(x_6) = 1$, ($D_A = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$), y B tal que $B(0) = 0.7$ ($D_B = \{0\}$). Entonces tenemos que el dominio de la erosión es en los dos casos $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$ y

$\xi_{g\text{-}I}(A, B)(x_i) = A(x_i) - B(0) = A(x_i) - 0.7$

$\xi_{g\text{-}I}(A, B) = \{-0.7, -0.5, -0.3, -0.1, 0.1, 0.2, 0.3\}$,

si utilizamos para la erosión borrosa la implicación $I(x, y) = \min(1, 1\text{-}x+y)$ nos queda

$\xi(A, B)(x_i) = I(B(0), A(x_i)) = \min(1, 1+A(x_i) - 0.7)$

$\xi(A, B) = \{0.3, 0.5, 0.7, 0.9, 1, 1, 1\}$

Si normalizamos estas imágenes al intervalo [0, 1] con la transformación T de $\xi_{g\text{-}I}$ (A, B) en [0, 1] tal que $T(x) = \frac{x - \inf \xi(A,B)(z)}{\sup \xi(A,B)(z) - \inf \xi(A,B)(z)}$, es evidente que una imagen tendrá 7 niveles de grises y la otra tendrá 6, las imágenes serán distintas.

3) Para demostrar que $\xi_{g\text{-}I}(A, B)(z) \in \{-1, 0\}$ para todo z perteneciente a D_f D_g, basta tener en cuenta que

$\xi_{g\text{-}I}(A, B)(z) = \inf_{x \in D_B \cap D_A - z} \{A(z+x)\} - 1$ y que A es un conjunto ordinario. Además se verifica:

$\xi_{g\text{-}I}(A, B)(z) = -1 \Leftrightarrow \inf_{x \in D_B \cap D_A - z} \{A(z + x)\} = 0 \Leftrightarrow$ si $B+z \not\subset A$

Y recíprocamente, si $B+z \subseteq A$, $\forall x \in B$ se verifica $(x + z) \in B+z \subseteq A$ y por tanto $A(x + z) = 1$

$\xi_{g\text{-}I}(A, B)(z) = 0 \Leftrightarrow \inf_{x \in D_B \cap D_A - z} \{A(z + x)\} = 1 \Leftrightarrow$ $B+z \subseteq A$.

Se recupera la erosión binaria.

Ver en anexo A los diferentes efectos producidos al erosionar una imagen borrosa con un elemento estructurante borroso utilizando el operador clásico para funciones "grey-level" y los operadores borrosos.

3. DILATACIONES

Podemos ver, que un sencillo cambio de variable permite escribir (6) en la forma

$$\mathcal{D}(A, B)(z) = \sup_{x \in D_B, x+z \in D_A} \{C(B(x), A(x + z))\} \quad (11)$$

Proposición 4.-

Si A y B son subconjuntos borrosos de D_A y D_B respectivamente, entonces

$$\mathcal{D}(A, B)(z) = \sup_{x \in SopB, x+z \in D_A} \{C(B(x), A(x + z))\}$$

Demostración.- La dilatación es operación dual de la erosión verificandose que $\mathcal{D}(A, B)(z) = N(\xi(N(A), B)(z))$ [2, 3, 5, 6-9]. Por ello, podemos escribir

$$\mathcal{D}(A, B)(z) = N(\inf_{x \in D_B \cap D_A - z} \{I(B(x), N(A(x + z)))\})$$

Por la Proposición 1, tenemos que

$$\mathcal{Q}(A, B)(z) = N(\xi(N(A), B)(z)) =$$

$$N(\inf_{x \in D_B \cap D_A - z} \{(I(B(x), N(A(x+z))))\})$$

$$\text{Pero } N(\inf_{x \in SopB \cap D_A - z} \{(I(B(x), N(A(x+z)))\}) =$$

$$\sup_{x \in SopB \cap D_A - z} \{N(I(B(x), N(A(x+z)))\})$$

Y teniendo en cuenta la definición del operador C, nos queda $\mathcal{Q}(A, B)(z) = \sup_{x \in SopB \cap D_A - z} \{C(B(x), A(x+z))\}$. .

Proposición 5.-

Si A es un subconjunto borroso de D_A y B un subconjunto Borroso de D_B, tal que $B(x) = K\ \forall x \in SopB$, entonces

$$\mathcal{Q}(A, B)(z) = C(K, \sup_{x \in SopB} \{A(x+z)\})$$

y su rango de pertenencia es

$$[C(K, \inf_{x \in SopB} \{A(x+z)), C(K, \sup_{x \in SopB} \{A(x+z)\})]$$

Demostración.- Por el apartado 2 de la Proposición 2 y teniendo en cuenta que $B(x) = K \in [0, 1]\ \forall x \in SopB$, se verifica

$$\mathcal{Q}(A, B)(z) = N(I(K, \inf_{x \in SopB \cap D_A - z} \{N(A(x+z))\}) \text{ y}$$

$$N(I(K, \inf_{x \in SopB \cap D_A - z} \{N(A(x+z))\}) =$$

$$N(I(K, N(\sup_{x \in SopB \cap D_A - z} \{A(x+z)\})))$$

y como $C(a, b) = N(I(a, N(b))$, resulta

$$\mathcal{Q}(A, B)(z) = C(K, \sup_{x \in SopB \cap D_A - z} \{A(x+z)\})$$

cuyo rango es $[C(K, G_2), C(K, G_1)]$

$$\text{Siendo } G_1 = \sup_z \{\sup_{x \in SopB \cap D_A - z} \{A(z+x)\}\} \text{ y}$$

$$G_2 = \inf_z \{\sup_{x \in SopB \cap D_A - z} \{A(z+x)\}$$

En particular si B es un elemento estructurante binario y teniendo en cuenta que $\forall b \in [0, 1]$ se verifica $C(1, b) = b$, tenemos que

$$\mathcal{Q}(A, B)(z) = C(1, \sup_{x \in SopB \cap D_A - z} \{A(x+z)\})$$

$$= \sup_{x \in SopB \cap D_A - z} \{A(x+z)\} \in [G_2, G_1]. \quad .$$

Veamos a continuación algunas coincidencias y diferencias entre los operadores dilatación de la erosión para funciones con tonos de gris y la dilatación borrosa definida anteriormente incluso limitándonos a utilizar elementos estructurantes centrados en el origen ($B(x) = B(-x)$).

Si se hace un sencillo cambio de variable la dilatación para funciones "grey-level" (4) puede escribirse

$$\mathcal{Q}_{g\text{-}I}(A, B)(z) = \sup_{x \in D_B, x+z \in D_A} \{A(z+x) + B(x)\} \quad (10)$$

Proposición 6.-

1) Existen subconjuntos borrosos A y B para los que la dilatación para funciones $\mathcal{Q}_{g\text{-}I}(A, B)(z)$ y la erosión borrosa no coincide $\mathcal{Q}(A, B)(z)$.

2) Si A, B son subconjuntos borrosos de D_A y D_B respectivamente, tal que $B(x) = K\ \forall x \in sop\ B$, $K \in [0, 1]$, entonces:

$$\mathcal{Q}_{g\text{-}I}(A, B)(z) = \sup_{x \in SopB, x+z \in D_A} \{A(z+x)\} + K$$

El dominio de $\mathcal{Q}_{g\text{-}I}(A, B)$ es

$$D_{\mathcal{Q}_{g\text{-}I(A, B)}} = [K, \sup_z (\sup_{x \in SopB, x+z \in D_A} \{A(z+x)\}) + K].$$

En particular si K = 1, la dilatación para funciones coincide con la dilatación borrosa.

Demostración.-

1) Veamos con un ejemplo que existen A, B subconjuntos borrosos de D_A y D_B respectivamente tal que ratifican que en general $\mathcal{Q}_{g\text{-}I}(A, B)(z) \neq \mathcal{Q}(A, B)(z)$. Sea A tal que $A(x_0) = 0$, $A(x_1) = 0.2$, $A(x_2) = 0.4$, $A(x_3) = 0.6$, $A(x_4) = 0.8$, $A(x_5) = 0.9$, $A(x_6) = 1$, ($D_A = \{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$), y B tal que $B(0) = 0.7$ ($D_B = \{0\}$). Entonces tenemos que el dominio de la dilatación es en los dos casos $\{x_0, x_1, x_2, x_3, x_4, x_5, x_6\}$ y

$$\mathcal{Q}_{g\text{-}I}(A, B)(x_i) = A(x_i) + B(0) = A(x_i) + 0.7$$

$$\mathcal{Q}_{g\text{-}I}(A, B) = \{0.7, 0.9, 1.1, 1.3, 1.5, 1.6, 1.7\},$$

si utilizamos para la dilatación borrosa la t-norma $C(x, y) = \max(0, x+y-1)$ correspondiente a la implicación $\min(1, 1-x+y)$ nos queda

$$\mathcal{Q}(A, B)(x_i) = C(B(0), A(x_i)) = \max(0, A(x_i) - 0{,}3)$$

$$\mathcal{Q}(A, B) = \{0, 0, 0.1, 0.3, 0.5, 0.6, 0.7\}$$

Si normalizamos $\mathcal{Q}_{g\text{-}I}(A, B)$ al intervalo [0, 1], con la transformación mencionada anteriormente, es evidente que una imagen tendrá 7 niveles de grises y la otra tendrá 6, las imágenes serán distintas.

2) Si A es una aplicación de D_A en [0, 1] y B una aplicación de D_B en $\{0, K\}$, $0 < K \leq 1$, entonces

$$\mathcal{Q}_{g\text{-}I}(A, B)(z) = \sup_{x \in SopB, x+z \in D_A} \{A(z+x)+K\} =$$

$$\sup_{x \in SopB, x+z \in D_A} \{A(z+x)\} + K, \text{ y } \forall z \in D_{\mathcal{Q}_{g\text{-}I(A, B)}} \text{ se}$$

verifica que $\mathcal{D}_{g\text{-}1}$ (A, B)(z) pertenece a

$$[K, \sup_z (\sup_{x \in SopB, x+z \in D_A} \{A(z+x)\}) + K]$$

En particular si K = 1,

$$\mathcal{D}_{g\text{-}1}(A, B)(z) = \sup_{x \in SopB, x+z \in D_A} \{A(z+x)\} + 1$$

y si normalizamos al intervalo [0, 1], la dilatación para funciones coincide con la dilatación borrosa, ver proposición 5. .

4. CONCLUSIONES

Para elementos estructurantes binarios la erosión, dilatación borrosa y las correspondientes operaciones para funciones "grey-level" coinciden salvo un factor de normalización al intervalo [0, 1]

$\xi(A, B)(z) = \xi_{g\text{-}1}(A, B)(z) + 1$. $\mathcal{D}(A, B)(z) = \xi_{g\text{-}1}(A, B)(z) - 1$.

Pero para imágenes y elementos estructurantes borrosos dichos valores aún normalizando al intervalo [0, 1], no coinciden, obteniendo efectos distintos en el procesamiento de imágenes.

5. REFERENCIAS

[1] D. Dubois y H. Prade, Fuzzy Sets y Systems: Theory and Applications, Mathematics in Science y Engineering, (Academic Press, New York 1980).

[2] De Baets, Bernad, Fuzzy morphology: a logical approch, Uncertainty Analysis in Engineering and Sciences: Fuzzy Logic, Statistics and Neural Network Approach (B. Ayyub and M. Gupta, eds.), Kluwer Academic Publishers, 1997, pp. 53-67.

[3] J. Serra, Image Analysis and Mathematical Morphology, Vol. 1 y 2, Academic Press, London 1982.

[4] Haas A., Matheron G. and Serra J., Morphologie Mathématique et granulométrie en place, Ann, Minos XI, 736-753 et XII, 1967, 767-782.

[5] N. Frago, Morfología Matemática Borròsa basada en operadores generalizados de Lukasiewicz: Procesamiento de imagenes. Ph.D Thesis, Universidad Pública de Navarra, Spain 1996.

[6] S. Divyendu y E. R. Dougherty, Fuzzification of Set inclusion, SPIE Vol. 1708 Applications of Artificial Intelligence X: Machine Vision and Robotics (1992) 440-449.

[7] S. Divyendu y E. R. Dougherty, Characterization of Fuzzy Minkowski Algebra, SPIE Vol. 1769 Image Algebra and Morphological Image Processing III (1992) 59-69.

[8] S. Divyendu y E. R. Dougherty, Fuzzification of set inclusion: Theory y applications, Fuzzy Sets and Systems 55 (1993) 15-42.

[9] S. Divyendu y E. R. Dougherty, Fuzzy Mathematical Morphology, Journal of Visual Communication and Image Representation, Vol. 3, No. 3, September 1992, 286-302.

[10] Smets P. and Magrez P., Implications in Fuzzy Logic. International Journal of Approximate Reasoning, 1 (1987) 327-347.

[11] Trillas E. and Valverde L., On implicatión and indistinguishability in the setting of Fuzzy logic, in: J. Kacprzyk, R.R. Yager, Eds., Management Decision Support Systems Using Fuzzy Sets and Possibility Theory (Verlag TÜV Rheiland, Köln, 1985), 198-212.

[12] Trillas E. and Valverde L., On inference in fuzzy logic, Proc. 2nd IFSA Congress, Tokyo, Japan, (1987) 294-297.

[13] Trillas E. and Valverde L., On mode and implication in approximate reasoning, in: Aproximate Reasoning in Expert System (M. M Gupta et al., Eds.), North-Holland, Amsterdam, (1985) 157-166.

[14] Trillas E. and Valverde L., On some functionally expressable implications for fuzzy set theory, Proc of the 3rd Inter Seminar on Fuzzy Set Theorhy, Linz Austria (1981) 173-190.

[15] P, Burillo,. N. Frago, R. Fuentes, Generación de Morfologías Matematicas Borrosas, Actas del VIII Congreso Español sobre Tecnologías y Lógica Fuzzy, Pamplona 1998. Páginas .

ANEXO A

En la Tabla 1 se muestran los efectos de la erosión borrosa con distintas implicaciones borrosas y la erosión clásica para este tipo de imágenes. Se usan distintos elementos estructurantes 3x3, definidos por B(i, j)=α, i, j∈{1,2,3} y α= 1, 0.7, 0.4. La imagen original es la Figura 1. Las rutinas para comprobar los efectos visuales se realizaron con la Aplicación Aphelion.

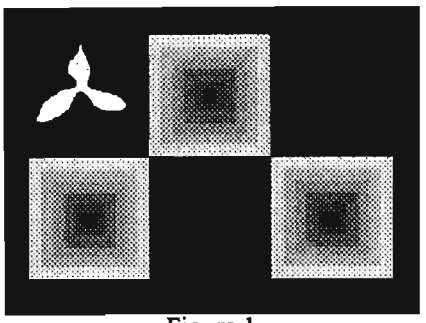
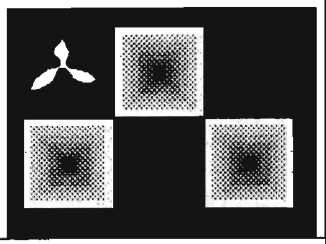
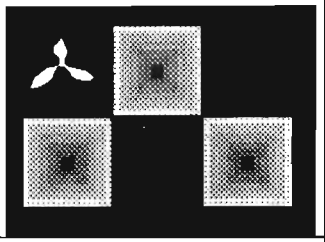
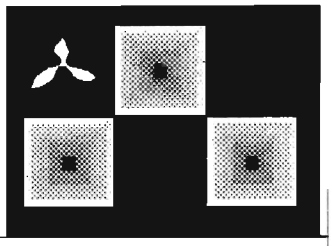
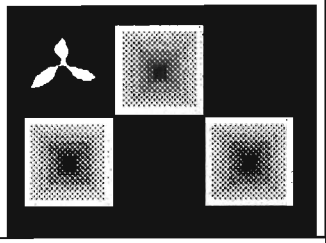
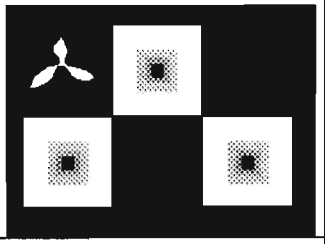
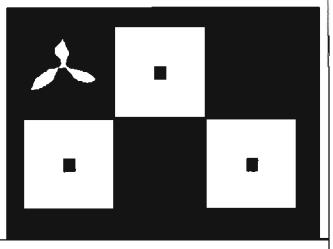
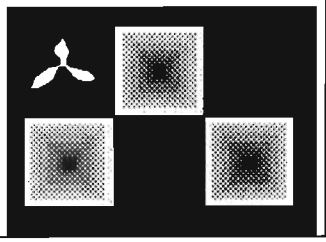
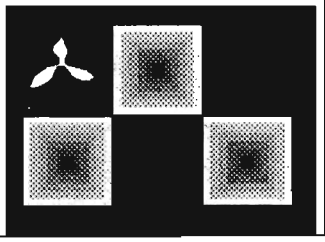
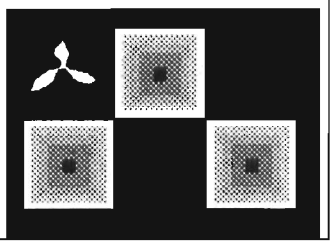


Figura 1

Tabla 1

	B(i, j)=1, i, j∈{1,2,3}	B(i, j)=0.7, i, j∈{1,2,3}	B(i, j)=3, i, j∈{1,2,3}
Erosión G-L clásica			
Erosión B, I. Lukasiwiec			
Erosión B, I. Klene-Dienes			

UN MODELO EVOLUTIVO DE OPTIMIZACIÓN DE PARTICIONES BASADO EN POLÍTICAS PREDICTIVAS

SILVA A.M.
Dep. Informática
Escuela Politécnica
Univ. Extremadura
10071 Cáceres (Spain)
agua@unex.es

LEÓN-ROJAS J.M.
Dep. Matemáticas
Escuela Politécnica
Univ. Extremadura
10071 Cáceres (Spain)
jmleon@unex.es

MASERO V.
Dep. Informática
Escuela Politécnica
Univ. Extremadura
10071 Cáceres (Spain)
vmasero@unex.es

MORENO J.
Dep. Informática
Escuela Politécnica
Univ. Extremadura
10071 Cáceres (Spain)
josemore@unex.es

Resumen

El presente trabajo pretende mostrar un método para optimizar la bondad de clasificación de una cierta partición difusa dada, construida sobre el referencial de medida de un sensor cualquiera, con el objeto de mejorar de forma evolutiva el ajuste de los atributos expresados mediante dicha partición con los datos observables del sensor. Para ello se utilizan políticas predictivas de evolución temporal de las medidas, basadas en la inferencia difusa de información cinemática y dinámica de las mismas. Usando técnicas de comparación entre los subconjuntos difusos estimados y los resultantes de clasificar un dato del sensor según la actual partición, el modelo es capaz de minimizar los errores cometidos en dicha clasificación mediante el reajuste de los puntos de definición de las funciones de pertenencia de los atributos clasificados por la partición vigente.

Palabras Clave: Optimización, Predicción, Particiones Difusas, (φ,ϕ)-Divergencias, λ-Divergencias.

1 INTRODUCCIÓN

En los problemas de clasificación mediante técnicas fuzzy se hace uso de particiones difusas del referencial de medida de la magnitud o característica a clasificar. Para que dicha clasificación obtenga buenos resultados dicho sistema de partición debe ajustarse a las agrupaciones de datos dentro de su referencial.

Los métodos de búsqueda de buenas particiones que satisfagan ciertas condiciones para ser usados en clasificación son variados y difíciles de desarrollar [3].

El método deseable sería aquel que se adaptara a los datos medidos y evolucionara hacia una solución que, si no óptima, fuera por lo menos subóptima [1] [2]. Pero este tipo de método de optimización requiere de mecanismos bastante complejos. Habitualmente se reduce dicha complejidad partiendo de un volumen de información a priori muy extenso.

Para intentar solucionar estas dificultades, proponemos un modelo que, partiendo de una partición genérica, sea capaz de evolucionar a una partición optimizada que clasifique correctamente las tendencias de agrupamiento de ciertos atributos en el referencial de medida de una magnitud concreta.

El modelo descrito se experimenta y verifica usando ejemplos de sensorización de movimientos propios de la mano humana para el reconocimiento de gestos conocidos, como pueden ser los signos de un lenguaje de sordomudos [8].

El planteamiento del modelo se basa en técnicas de comparación entre subconjuntos difusos, las cuales miden diferencias entre las funciones de pertenencia de los mismos, tanto en un sentido *horizontal* [4] como en uno *vertical* [5], es decir, sobre sus soportes y sobre la distribución de sus alturas.

La forma de realizarlo es obteniendo una estimación de la futura medida, o sea, una predicción de la respuesta del sensor, basada en la inferencia difusa de información cinemática y dinámica de las observaciones anteriores. Ésta se compara con el resultado difuso de clasificar el dato real observado del sensor según la actual partición, obteniéndose unas medidas de la bondad del clasificador.

Con esa metainformación, el modelo es capaz de minimizar los errores cometidos mediante el reajuste de los puntos de definición de las funciones de pertenencia de los atributos clasificados por la partición vigente.

Para que este método de optimización sea factible, el modelo de predicción debe ser robusto, por lo que tenemos que exigir que la adquisición de los datos deba realizarse con una frecuencia constante y suficientemente alta para simular un estudio continuo de las mismas. Esto hará que las diferencias entre estimación y medida sean debidas únicamente a desajustes de los atributos de la partición vigente y no a fallos en la obtención de los datos del sensor.

2 MODELO PROPUESTO

2.1 PARTICIONADO DIFUSO

Partimos de una partición completa genérica del referencial de medida de un sensor, aunque podemos incluir en esta partición de partida información obtenida por un a priori estadístico redefiniendo el rango efectivo del sensor.

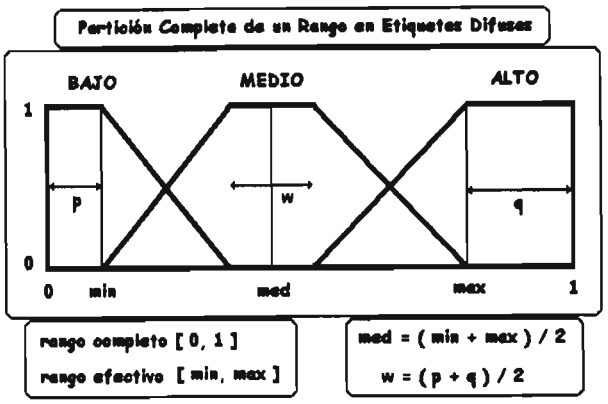


Figura 1: Ejemplo de Partición Difusa.

Utilizamos la siguiente forma de definición de las funciones de pertenencia de los atributos a partir de ciertos puntos como el punto *medio* del núcleo y su *radio*, las amplitudes laterales izquierda y derecha. Haciendo salvedades en los atributos extremos, donde alguno de estos puntos de definición no existen o están desvirtuados.

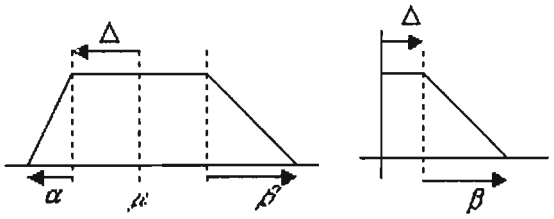


Figura 2: Definición de Atributos.

2.2 ESTIMACIÓN PREDICTIVA

La predicción de la medida del sensor se obtiene por inferencia a través de la relación difusa generada entre la estimación de la velocidad y la estimación de la aceleración[1]. La predicción resultante de dicha inferencia es un subconjunto difuso donde los parámetros: μ, Δ, α y β, vienen dados por el resultado de clasificar la estimación de la medida según la partición existente y la posterior modificación de esa cantidad difusa según el nivel de asimetría (creencia en la estimación) obtenido en la relación de la tabla siguiente.

Tabla1: Inferencia Cinemato-Dinámica.

VELOCIDAD ESTIMADA \ ACELERACIÓN ESTIMADA	IB	IM	IS	Z	DS	DM	DB
IB			LIB		LIM	LIS	
IM	LIB			LIM			SIM
IS	LIM		LIS		LIS	SIM	
VZ	LIS			SIM			LDS
DS	LIS	SIM			LDS		LDM
DM	SIM		LDS		LDM		LDB
DB	SIM	LDS	LDM		LDB		

L *es Asimetría* I *es Izquierda* D *es Derecha*
S *es Pequeña* M *es Mediana* B *es Grande*

2.3 COMPARACIÓN DE DIFUSOS

La comparación entre subconjuntos difusos se realiza usando técnicas de cálculo de divergencias entre ellos, en las cuales se buscan diferencias entre las funciones de pertenencia de los mismos, tanto en un sentido *horizontal* [6] como en uno *vertical* [7], es decir, sobre sus soportes y sobre la distribución de sus alturas.

Los resultados de estas técnicas nos permiten ajustar los parámetros de definición de las funciones de pertenencia de los distintos atributos implicados en la comparación anterior. Esto nos ayuda a mejorar el modelo de predicción, es decir, ayuda a aumentar la bondad del clasificador que está usando dicha partición [10].

2.4 REORGANIZACIÓN DE ATRIBUTOS

El método anteriormente descrito de redefinición de funciones de pertenencia puede orinar modificaciones tanto verticales como horizontales, siendo las primeras debidas a variaciones del núcleo *(intra-atributo)* y las últimas a variaciones del soporte *(interatributos)*.

El primer tipo se da cuando el resultado de la comparación ofrece divergencias en la predicción y estimación que afectan mayoritariamente a un solo atributo, es decir, se ve afectado únicamente el núcleo del atributo en cuestión, aunque por el tipo de partición usada, se modifican parte de los soportes no pertenecientes al núcleo de los atributos adyacentes para

[1] Para estos cálculos se usan aproximaciones polinómicas de la primera y segunda derivada discretas de la evolución temporal de la medida.

seguir siendo perfecta[2]. La interpretación de estos cambios nos informa de variaciones en la dispersión o concentración de la clase representada.

Por otro lado, las variaciones interatributos son debidas a divergencias que implican a más de un atributo. Este tipo de variaciones afecta a la distribución de los soportes de dichos atributos.

La interpretación de este tipo de redefiniciones es más compleja, pues absorbe el primer tipo afectando también a los soportes de los vecinos adyacentes. Su significado es la reorganización de las clases representadas, subsanando errores previos de mala clasificación, es decir, acepta como pertenecientes válidos de clase a anteriores *outlayers*, o por el contrario, marca como tales a anteriores valores erróneamente aceptados como válidos.

Esta reorganización de la partición difusa tiene dos tipos de singularidades debidas a crecimiento y encogimiento extremo del soporte de los atributos. Estas singularidades pueden originar la generación de nuevos atributos por *separación* de uno muy extenso, y la desaparición por *absorción* de uno muy estrecho, respectivamente. Siendo esta última un problema para la robustez del método, pues puede originar *fusiones* entre clases.

Para tratar estas singularidades hay dos vías: evitarlas introduciendo límites asintóticos al crecimiento del núcleo y a la reducción del soporte, o asumirlas como parte inherente del modelo, ampliando este último con ciertas restricciones para evitar el efecto colateral de las fusiones en cascada. Estas restricciones se incluirán como recorte de los núcleos adyacentes cuando se produzcan absorciones.

3 EXTENSIÓN DEL MODELO AL ESTUDIO DE REFERENCIALES MULTIDIMENSIONALES

La extensión del modelo anterior a estudios donde las particiones difusas se realizan sobre referenciales multidimensionales se puede hacer generando nuevos modelos de divergencias que midan disimilitudes globales en esos mundos multidimensionales, o por el contrario, trabajando con proyecciones sucesivas de dichos referenciales y realizando optimizaciones locales a una única medida.

La solución más satisfactoria es sin duda la primera, pero la complejidad se dispara al tratar de generar estudios de divergencias multidiemensionales de forma que puedan medir disimilitudes *horizontales* y *verticales* para unos niveles de vecindad que nos desbordan.

La solución basada en proyecciones sucesivas tiene la ventaja de su relativamente *baja* complejidad. Pero predecir basándose en estimaciones sobre las proyecciones tiene asociado una pérdida credibilidad bastante fuerte desde el punto de vista de la interpretación de las clases, ya que se pierde casi totalmente la no-linealidad del comportamiento de los atractores de clase.

[2] También llamada completa. La suma de las funciones de pertenencia de todas las partes es uno para todo valor del referencial de medida.

4 OTRO TIPO DE EXTENSIÓN MULTIDIMENSIONAL: JERARQUIZACIÓN DEL MODELO

La extensión del modelo a estudios donde las particiones difusas se definen sobre referenciales multidimensionales ocasionan como se ha comentado previamente pérdidas de efectividad y/o eficiencia en la búsqueda de la optimización, así como un incremento considerable de la complejidad.

Para intentar compensar las dos vías anteriores de solución, proponemos un modelo híbrido donde se intente aprovechar la sencillez y elegancia del modelo proyectivo, así como la potencia de búsqueda de óptimos globales del otro modelo.

Esta extensión se basa en la distribución jerárquica [9] del referencial multidimensional. De este modo conseguimos trabajar con modelos que son capaces de extender el método unidimensional a otro bidimensional o tridimensional, *fáciles* de resolver.

Para extensiones a más dimensiones se utilizan las optimizaciones locales de los referenciales bi y tridimensionales pasándose a tratar éstas particiones como una única dimensión, absorbiéndose consecutivamente por un procesamiento secuencial las restantes dimensiones de orden superior en jerarquía, y por un procesamiento paralelo las informaciones de orden paritario.

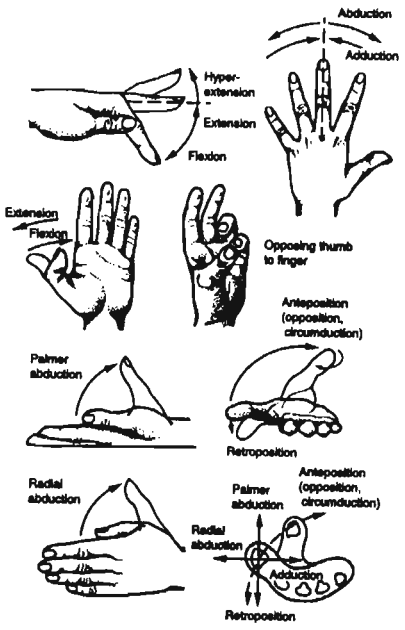


Figura 3: Ejemplo de Aplicación Jerarquizada. Reconocimiento de Gesticulación Manual.

Agradecimientos
Los autores desean expresar su gratitud a la Comisión Interministerial de Ciencia y Tecnología, así como a la Junta de Extremadura, por la ayuda recibida a lo largo de varios proyectos: PTR-94-0051 y PTR-95-0031, y PRI-97-C1063 y PRI-98-C006, respectivamente, lo cual nos ha permitido financiar parcialmente nuestra investigación.

Referencias

[1] O. Cordón y F. Herrera. A Hybrid Genetic Algorithm-Evolution Strategy Process for Learning Fuzzy Logic Controller Knowledge Bases. *Fuzzines and Soft Computing,* 8, 251-278, 1996.

[2] B. Filipic y D. Juricic. A Genetic Algorithm to Support Learning Fuzzy Control Rules from Examples. *Fuzzines and Soft Computing,* 8, 403-418, 1996.

[3] H. Genther y M. Glesner. Advanced Data processing using Fuzzy Clustering Techniques. *Fuzzy Sets and Systems,* 85, 155-164, 1997.

[4] J.M. León-Rojas, J. Moreno, A.M. Silva y M. Morales. Una Familia de Divergencias entre Conjuntos Borrosos. *XXV CNEIO,* 797-798, 2000.

[5] J.M. León-Rojas, J. Moreno, A.M. Silva y M. Morales. Malla de Divergencias entre Conjuntos Borrosos. *XXV CNEIO,* 795-796, 2000.

[6] J.M. León-Rojas, A.M. Silva, J. Moreno y M. Morales. Applications of the (φ,ϕ)-Divergences between Φ-Probabilistics Sets in Hand Motor-Skills Evaluation and Rehabilitation Processes. *METMBS,* 2000.

[7] J.M. León-Rojas, J. Moreno, A.M. Silva y M. Morales. A Family of Divergences between Φ-Probabilistics Sets with Appications to Handshape Recognition. *IAPR—SSSPR,* 2000.

[8] J. Moreno, J.M. León-Rojas y A.M. Silva. Sistema de Traducción Automática del Lenguaje de Signos Español al Español Oral. *Informática y Discapacidades. ATI Novática*, 136, 1998.

[9] A.M. Silva, J. Moreno, J.M. León-Rojas y F. Andrés. Hierarchical Hand Gesture Recognition model for Virtual Reality Applications in Medicine. *METMBS,* 2000.

[10] A.M. Silva, J. Moreno, J.M. León-Rojas y V. Masero. A Dynamic Approach to Human Gesture Recognition. *IAPR—AMDO,* 2000.